the TOP keyword in SQL

Posted At : March 18, 2005 3:34 PM | Posted By : Mark Kruger
Related Categories: SQL tips, Coldfusion & Databases

One of the most common things I see when looking at code I'm trying to optimize is a missunderstanding of how to effectively utilize SQL and a good database platform. Nothing illustrates this more effectively than the constant use of the "maxrow" attribute for a `<cfquery>` or a `<cfoutput>` statement. In many cases the **TOP** keyword should be used instead. Let me Explain...

h4>When to use "maxrows"

Before we talk about *TOP* let me say that *maxrows* are a good choice for solving a number of problems. However, I can think of almost no cases where "maxrows" should be used in a `<cfquery>` tag. Passing the *TOP* keyword as a part of the select will accomplish the same thing, but it will allow the database server to develop the execution plan appropriately - instead of the CF server and ODBC driver. There are some cases where *maxrows* can benefit your `<cfoutput>` or `<cfloop>` statements. 2 rules of thumb to follow however:

1. The total number of records in the recordset should be limited. The limit will vary depending on your system resources.
2. The query has been optimized with appropriate "where" clauses.

In regard to item 1, I've often seen sites where ***ALL*** the records in a table where pulled into the record set, then then the top 10 or 15 records are displayed. I saw one case where a site was displaying news for example. Each request would pull in 2 or 3 *hundred* stories (SELECT * ... titles, body, meta-data and all) ordered by date and then display only the headlines of the latest 10 or 12 stories. This is basically a waste of memory and network bandwidth.

As for Item 2, the rule of thumb for *ANY* query is "..in as much as possible, only retrieve what you will need for this request." Use a date, or filter or subquery - whatever you can - to trim the query down to just what you need. In the case above, simply adding a where clause that exclude news stories many weeks old would have been a vast improvement - even without the "top" keyword.

Typical Use

You want a *positive* example? Sure... how about a typical e-commerce web site. Let's say you are going to display the product page for a category. Perhaps you have 50 products, but you only want to display 10 on each page. Pulling all 50 into a *cached* query makes sense because it is 1 trip to the database for 50 records as opposed to 5 trips for 10. Then, you use "maxrows" to make your "next/prev" interface. The first page would use something like this:

```
<cfoutput maxrows="10" startrow="1">
   output here and a "next" link...
      .....
</cfoutput>
```

...And the second page something like this:

```
<cfoutput maxrows="10" startrow="11">
   output here and a "next" and "prev" link
```

```
      .....
</cfoutput>
```

and so on. This is exactly the sort of use of maxrows that will benefit performance and make your site more intuitive as well.

TOP is king

Ok... so maxrows has it's uses, but what if you have a bazzilion records? Or... what if you have a table that is fine now, but you expect it to grow over time. That's a fairly common mistake. You start out with code that works fine, but as the table grows larger your application bogs down because you haven't accounted for the recordset sizes. In that case the keyword "TOP" comes to the rescue. It's easy to use. It follows SELECT and then it is followed by a number specifying how many records maximum you want to retrieve. So "SELECT TOP 10 * FROM products returns only 10 records.

Let me emphasise that this is *different* from using `<cfquery maxrows="10">`. In the second case, the ODBC driver handles passing the query (the WHOLE query - select * from products) to the RDBMS - which first runs the query (getting all the records) and then begins to pass them back using the blockfactor settings. In each case, the RDBMS is building a recordset that is larger than needed and most certainly passing back MORE records than 10 to the ODBC driver - which passes them to CF. CF in turn, looks for 10 or more records and terminates the query when it has the required amount - passing those first 10 received back to the request. If you don't believe me, do a trace and you will see the nature of the request being passed to the RDBMS.

With TOP, the RDBMS creates an execution plan based on the where clause *and* the "TOP" keyword - then begans building ONLY until it has the required number. At that point it stops execution and forwards the records back to the web server. Here's an example of TOP in action:

```
<cfquery name="getNews" datasource="myNews">
   SELECT TOP 25 Title, FileDate, Byline
   FROM   tblCurrentNews
   WHERE   FileDate > DateAdd(day,-1,getdate())
</cfquery>
```

Notice that TOP precedes any specified columns. Top can be used with any combination of "GROUP BY" or "UNION" or "JOIN". The only limitation is that some views that contain inner joins cannot be created correctly using TOP. Also notice that I'm letting the database server do the job of creating a date that is 1 day earlier than now. This is much better than using the "now( )" function with "createOdbcdate( )" and the CF "dateadd( )" function.